Prototyping a Multi-Root ONS

Roberto Quilez, Nathalie Mitton, Marcelo Dias de Amorim, Nicolas Pauvre

Prototyping a Multi-Root ONS

Roberto Quilez*, Nathalie Mitton*, Marcelo Dias de Amorim◇, and Nicolas Pauvre†

* INRIA ◇ CNRS †GS1

***Abstract*—The Object Naming System (ONS) is a central lookup service used in the EPCglobal network for retrieving location information about a specific Electronic Product Code (EPC). This centralized solution lacks scalability and fault tolerance and encounters some political issues. We present the design principles of a fully-distributed multi-root solution for ONS lookup service. In distributed systems, the problem of providing a scalable location service requires a dynamic mechanism to associate identification and location. We design, prototype, and evaluate PRONS, a solution based in a distributed hash table (DHT) for the multi-root problem. We show that PRONS achieves significant performance levels while respecting a number of neutrality requirements.**

I. Context and motivation

As we move forward towards ambient intelligence environments where most devices are connected to seamless, ubiquitous networks, inter-enterprise interoperability becomes an essential prerequisite. Integrated complex networks, composed of a huge amount of different types of objects, form the so-called Internet of Things [1]. A subcategory of these networks, known as the Internet of Goods will manage information exchanges of a networked business-to-business world. The architecture supporting this increase in scale should be designed accordingly to follow an open governance model.

The EPCglobal network will be part of the Internet of Things.[1] EPCglobal is a subscriber-driven organization comprised of industry leaders and organizations focused on creating global standards for the EPCglobal Network. EPCglobal has developed a collection of interrelated standards for hardware, software, and data interfaces, together with core services that are operated by its delegates. One of its key standard-based components is a centralized objects directory service called the ONS (Object Naming Service) [2], which is based upon the DNS (Domain Name System) [3]. The ONS defines the interface for lookup services by providing quasi-permanent or relatively static links between the identity of a company responsible for an object (often the manufacturer) and the authoritative information services which that company provides. This company has a database made of object relative data. This information is managed via the EPCIS (Electronic Product Code Information Services) of the network partners. Based on the Internet DNS protocol, the ONS is a hierarchical client/server model offering the possibility to orient requests coming from client applications towards the EPCIS of the right company.

ONS Standard version 1.0.1 designs the EPC network as a global information system, centralized, in which several local ONS are interconnected.[2] To respond to application queries by routing them towards the right local ONS, a root domain is implemented (onsepc.com). This root domain, or root-ONS, is the system core, structuring the network and localizing associated services. This is a unique and authoritarian root, as it refers in facto every local ONS of the EPCglobal network. The main problem today is exactly that the current implementation of the ONS considers a single centralized entity at the root level. There has been a number of initiatives worldwide that argue in favor of a completely neutral, distributed organization of the root level [4], [5]. In short, the question to be solved is how to deploy a *distributed governance system* on a per-country basis. Furthermore, the current centralized solution based in a single DNS server *lacks scalability and fault tolerance*. The emergence of multiple roots would require that the standard evolve toward a symmetric architectural model.

Among the requirements related to governance, the following have direct impact on the core functionalities of the system:

- There must be no central authority above the ONS roots. No ONS root detains a higher responsibility than another one. Each root is equal.
- The system must be robust to churn. The system must be scalable, robust to heavy load and support a potentially great number of roots or data.
- The system must be compatible with the current ONS implementation of GS1.[3]
- The system should provide easy implementation of security solutions.

These points make the design of a solution challenging. In this paper, we design, prototype, and evaluate PRONS, a DHT-based substrate to organize a potentially large number of root nodes while respecting the requirement listed above. Other works have adopted similar strategies [5]. Contrarily to these solutions, our proposal is totally compliant with the existing architecture and focuses only on the root nodes.

II. Towards a multi-root ONS

In this section, we present our solution to the multi-root ONS. The basic idea behind the solution is to rely on the concept of distributed hash tables (DHT). The reason for such a choice, as it will become clearer in the remainder of this document, is that DHTs gather enough properties to cover all the requirements stated in Section I.

[1] http://www.epcglobalinc.org

[2] http://www.epcglobalinc.org/standards/ons/ons_1_0_1-standard-20080529.pdf

[3] http://www.gs1.com

A. Infrastructure

Root ONS servers are nodes on a DHT that manage a common addressing space. GS1 Member Organizations (MOs) or any other authoritative agent run a peer root ONS. They also serve as an entry point to the service for any of the implemented interfaces (Section III-C). In other words, each peer root ONS acts as a DHT node with a unique node identifier, produced by a SHA1 hash of the node's IP address.

ONS roots store couples as {key, value}. The GS1 Company Prefix will be used as key to index the location of the corresponding local ONS. Company prefixes provide a way for GS1 Member Companies to uniquely and globally identify things like trade items, logistic units, locations, parties, and assets and will be hashed in order to be mapped into into a position on the addressing space. Note that later on, we may use an other identifier as key (such as the EPC for instance) regarding the future needs. It will not consist in a huge change. Pointers (i.e., network addresses like DNS name or IP address) to corresponding local ONS are the values associated to keys. Note that the DHT implementation does not entail any restrictions concerning the format of the values, so other potential information associated to future services could be included as well.

Local ONS. The Local ONS will fulfill ONS lookup requests for EPCs within the control of the enterprise that operates it; that is, EPCs for which the enterprise is the EPC Manager.

B. Clients

Clients perform queries to the ONS. They use the information provided by an Electronic Product Code (EPC), assigned to an item, to request the system about associated information.

C. Bootstrapping

At least one node already in the federation must be provided to join the overlay network and/or benefit from the peer root services. Different solutions could be adapted according to the scenario we are considering.

If the peer root nodes are integrated in the DNS system as described in Section IV-A, bootstrapping nodes are reached by profiting of zone delegation. In the case of a DNS independent scenario new nodes could contact a bootstrapping server first and get a partial list of existing nodes. Another common approach is to let nascent nodes know in advance an entry point into the network (e.g., a list of known nodes of the overlay or a list of non-public bootstrapping servers) provided by GS1 for instance.

III. PRONS Architecture

We have developped a shell application called PRONS (PeerRootONS). It is command language interpreter which controls a higher-level service DHT. It invokes a node on a structured overlay implementing a root ONS federation and controls a DHT service according to user's instructions given from standard input. We now describe some of the implementation choices taken to develop the DHT based solution, its modular architecture at run time and the enabled interfaces to operate the application. We depict in as shown in Fig. 1 the overall architecture of the system. More detailed information on the architecture can be found in [6].

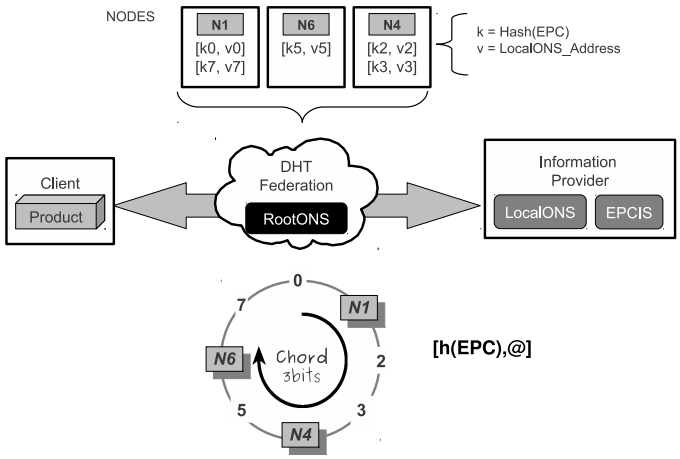

Fig. 1. Peer Root Object Name Service.

A. Technology: Choices and limitations

The basic operation of the PRONS application is to implement an overlay network based on Chord [7]. There are several Chord implementations available but no official release of the protocol [8], [7]. Since we are not implementing a solution to be deployed in very different user environments but for root servers, portability is not mandatory. Our implementation is done in Java for debugging reasons. This is not a big issue for the time being, although a more efficient implementation in C could be envisioned for the production system.

B. Modules

PRONS has been built over the common API for higher-level services provided by the Overlay Weaver toolkit [9]. It is an open source DHT software developed as a research project. It is highly modular, configurable, and customizable. Relying on this common API, PRONS does not depend on specific transport protocols, database implementation, or routing algorithms. Fig. 2 shows the components organizing at the runtime.

This multi-layer architecture of PRONS is structured in three main layers: application, high-level services, and routing and storing services. The routing layer corresponds to the key-based routing layer as proposed in [10] but it was split into three parts in the toolkit: routing driver, routing algorithm, and message service. This decomposition allows implementing a number of well known overlay algorithms only in about a hundred lines of code and test the performance of the system with different implementations. Additionally, the decomposition allows multiple implementations of the routing driver (the toolkit provides Iterative and Recursive) and Messaging Service (UDP, TCP).

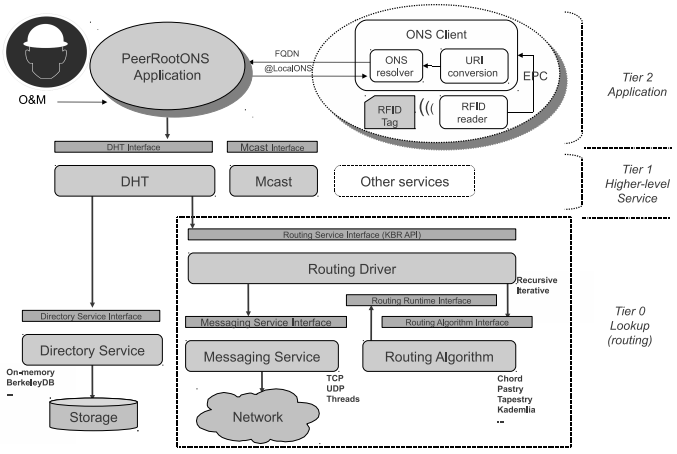

Fig. 2. PRONS application and OW modules at runtime.

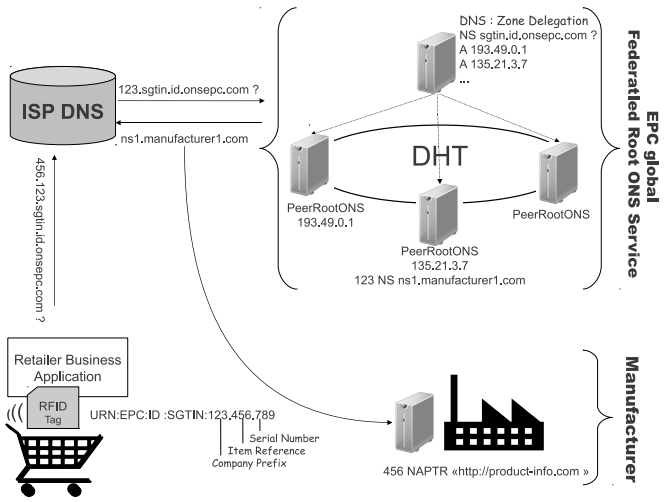

Fig. 3. DNS compliant peer root ONS solution.

C. Agents and Interfaces

Each node interacts with three main agents:

- **Authoritative agents:** As GS1 MOs, they will be allowed to perform operations on nodes (e.g., join new nodes to the federation) and objects. In other words, this means that they have to manage the information stored in the DHT (through operations like put and delete).
- **Clients:** They will query the root service asking for information associated to company prefixes.
- **Other root nodes:** DHT services.

In order to enable such interactions between agents and the peer root node, the following interfaces are defined:

- **PRONS shell (or remote shell server):** Command language interpreter that controls a higher-level service DHT.
- **XML/RPC and Web Service:** The interface by which clients can access the peer root ONS service using XML-RPC over HTTP. Section IV describes a sample scenario enabling an interaction between clients and root services [11]. On the same port, PRONS shell provides a web interface on which node information can be seen with a web browser.
- **DNS:** A basic DNS server interface to attend queries according to the specification [2]. Resource records are retrieved from the DHT instead of a local configuration zone file (see Section IV-A).

IV. Integration scenarios

We have defined the way the root nodes operate to store/retrieve the information internally but the client interface remains open. To define the client interaction we consider two different scenarios:

- The first one is compliant to the current standard [2] but takes advantage from the peer features.
- The second one where ONS root service could be implemented independently from the DNS service and protocol, accessing the roots in a more general manner. XML/RPC has been implemented to address it.

Note that there are no reasons that refrain these two scenarios to coexist.

A. DNS compliance

The ONS root service as specified in [2] is performed by a single DNS server. We propose to split this single server into a federation of peers implementing a DHT. Peer root nodes share the service governance and database administration.

For the purpose of integrating the DHT to the rest of the DNS system, the management of a DNS domain (`sgtin.id.onsepc.com.`) is delegated to a federation orchestrated by the DHT. Each node in the federation implements a DNS shell so that it is able to answer FQDN (Fully Qualified Domain Name) queries in the same way that current single ONS root does. Those nodes provide valid entry points to the DHT. The only difference, transparent from the client's point of view, is that the contacted node is able to efficiently retrieve the value (local ONS address) associated with a given key (company prefix) from the DHT instead of a local zone file as in the traditional DNS.

Zone Delegation. Authority over a portion of the DNS namespace `onsepc.com.` is assigned to the subdomain `sgtin.id.onsepc.com.` within this namespace. The responsibility for the resource records of the subdomain is passed from the owner of the parent domain to the owner of the subdomain.

In this particular case, as we can see in Fig. 3, the management of the `sgtin.id.onsepc.com.` domain is performed by the collection of nodes implementing a DHT. The administrator of the `sgtin.id.onsepc.com.` zone (i.e., the DHT administrator) controls the resource records for that subdomain. This zone delegation provides us the way to:

- Delegate the management of a DNS domain (`sgtin.id.onsepc.com.`) to a federation of peers (DHT).

- Balance the load of maintaining one large DNS database among multiple servers to improve object name resolution performance and fault tolerance.

For a delegation to be implemented, the parent zone must contain both an A resource record and an NS resource record pointing to the authoritative server of the newly delegated domain. These records are necessary both to transfer authority to the new name servers and to provide referrals to clients performing iterative queries.

Zone delegation must be done to DHT giving at least one peer root node acting as a gateway to the DHT. In order to provide a simple load balancing solution, multiple A records with the same name (`sgtin.id.onsepc.com.`) and multiple IP addresses (corresponding to peer root nodes) could be defined. In this way, all nodes in the DHT are potential points of entry. In this case the load-balancing effect is under the control of [12] (see `rrset-order named.conf` statement). The DNS delivers all IP addresses defined, the first IP address in the list being in a default round-robin distribution.

B. DNS independence

Adapting ONS requests to the DNS protocol and defining a service to use the root directory to find information about an item is a mere implementation choice that presents the limitations and consequent lack of flexibility of a service originally designed for other purposes. Given that we have implemented a distributed system providing a lookup service where pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key, using the DNS protocol is no longer mandatory. In this way, we also propose a DNS-independent scenario where queries to the root service are implemented in a more generic way.

Each peer root node serves as a gateway to the DHT. A gateway accepts RPC operations from clients, forwards those messages into the DHT, and forwards the corresponding responses to those operations from the DHT to the appropriate client. RPC is invoked using XML-RPC [11], which is a remote procedure that uses HTTP as the transport layer and XML as the encoding language. It has been designed to be as simple as possible, while allowing the manipulation of complex data structures.

DNS could still play a small role as a bootstrap mechanism to the DHT (Section II-C). In order to avoid this, a list of all active peer root node servers could be downloaded from a web server. This list could dynamically be updated to reflect the state of the peer root service deployment. The shortest latency for completion of peer root ONS RPCs will generally be experienced if a gateway topologically near to the client on the Internet is chosen. To find a nearby root ONS gateway a script could be also provided.

V. Evaluation

Three nodes with public IP addresses have been deployed in order to test the multi-root DHT approach at the qualitative (functionality) and quantitative (performance) levels. Each node runs an instance of the PRONS application. All nodes are connected through an overlay network implementing a DHT and providing XML/RPC and DNS interfaces as described in previous sections. The distributed hash table partitions the address space among the participating nodes. Each node has been mapped to the Chord addressing space by hashing its public IP address. We have generated keys from a random pattern of three characters followed by consecutive numbers. Those keys have been hashed (SHA1) and stored in the DHT.

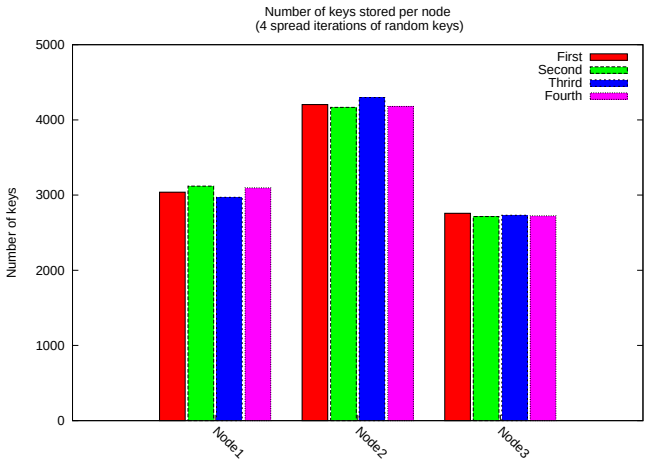

Fig. 4. Keys allocation.

TABLE I
NAMESPACE REPARTITION OVER NODES.

NODE ID	% Chord addressing space
`1be19...`	`31% aprox.`
`87709...`	`42% aprox.`
`ccf7b...`	`27% aprox.`

Fig. 4 shows the key allocation for four different sequences of patterns of $1,000$ keys each, while Table I depicts the namespace distribution over nodes. As a result of the consistent hash we can see that keys are distributed uniformly along the addressing space, so the load of a node strongly depends on the percentage of the Chord addressing space managed by that node. For a huge number of nodes fairness is ensured but a load-balancing problem arises for a reduced number of nodes so in this case their placement should be monitored.

For the XML/RPC scenario as a standalone solution (Section IV-B), independent from the DNS system, Fig. 5 shows the throughput results for the responses coming from the multi-root ONS federation giving the local ONS pointer for the requested company prefixes. In order to measure the throughput of the implemented solution, we have requested two different gateways of the DHT geographically distributed. The number of queries is equivalent to the frequency [queries/sec.] being used so they all should be attended in 1 second time. We observe that for frequencies higher than 100 [queries/sec.] the total delay becomes to rise over the expected limit.

In order to show the outcome of our evaluation steered towards measuring the latency penalty due to the use of PRONS, DNS requests have been performed enabling the tracing of the delegation path from the root name servers for

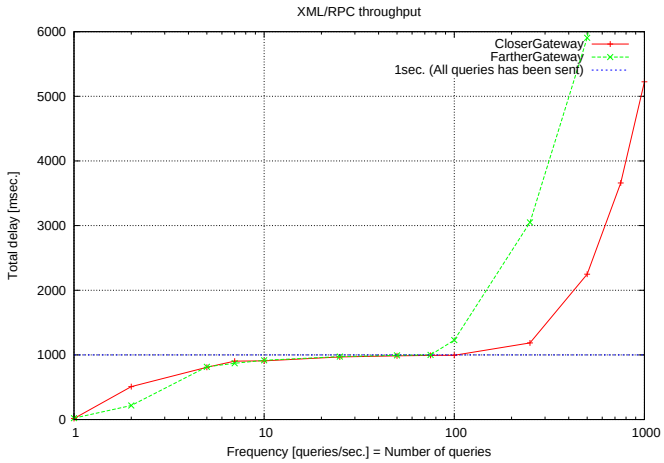

Fig. 5. XML/RPC throughput.

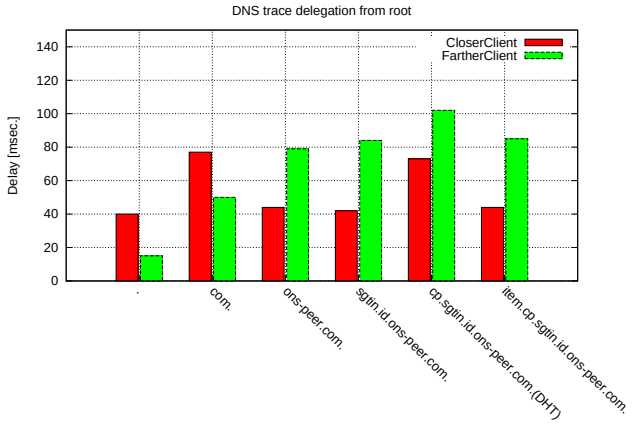

Fig. 6. Delay per DNS name server in the delegation path from the root name server until the local ONS for iterative queries.

the EPC being looked up until the NAPTR record stored in the corresponding local ONS. Fig. 6 shows the result of iterative queries made to resolve the EPC being looked up. They follow referrals from the DNS root servers till the local ONS, passing by the ONS root federation. The delay for the answer from each server that was used to resolve the lookups is displayed.

Note that the three DNS paths (item.cp.sgtin.id.ons-peer.com, sgtin.id.ons-peer.com, and ons-peer.com) get similar latency to reach one station. We can observe a lightly bigger latency for the response coming from the multi-root federation (i.e., cp.sgtin.id.ons-peer.com) which potentially goes through several stations in a transparent way to the DNS resolution system – this is reasonable considering that several stations are reached and still considerably smaller in comparison to the answer of some of the DNS servers.

VI. Summary and Outlook

In this paper, we proposed PRONS, a distributed ONS mechanism totally compliant with the current EPC Global ONS standard but that solves scalability and political issues. We have shown that PRONS may be used in a stand-alone way together with a traditional DNS-based fashion while still offering good performance levels. Next steps will be to analyze the key repartition over the overlay network in order to balance not only the data storing but also the client requests. Indeed, some products may be queried much more often than some other ones and thus, although each peer node is responsible for the same amount of data, they might not be all queried equally.

Acknowledgment

This work is partially supported by the ANR project WINGS under contract ANR-09-VERS-015.

References

[1] ITU, *Internet Reports 2005: The Internet of Things.* ITU, 2005.
[2] E. Global, “ONS standards,” http://www.epcglobalinc.org/standards/ons, 2008.
[3] DNS, “Domain name service,” http://www.howstuffworks.com/dns.htm.
[4] H. XU, S. WANG, and R. WANG, “P2PONS: A distributed object naming service architecture based on p2p for epc network,” *Advances in Information Sciences and Service Sciences (AISS)*, vol. 3, no. 3, pp. 1–10, 2011.
[5] B. Fabian and O. Gunther, “Distributed ONS and its Impact on Privacy,” in *International Conference on Communications (ICC)*, June 2007, pp. 1223–1228.
[6] M. D. D. Amorim, S. Fdida, N. Mitton, L. Schmidt, and D. Simplot-Ryl, “Distributed planetary object name service: Issues and design principles,” INRIA, Research Report 7042, 09 2009.
[7] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup protocol for internet applications,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, Feb. 2003.
[8] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, “Opendht: A public dht service and its uses,” in *Proc. of ACM SIGCOMM 2005*, 2005. [Online]. Available: http://www.opendht.org/
[9] K. Shudo, Y. Tanaka, and S. Sekiguchi, “Overlay weaver: An overlay construction toolkit,” *Computer Communications (Special Issue on Foundations of Peer-to-Peer Computing)*, vol. 31, no. 2, pp. 402–412, 2008.
[10] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica., “Towards a common api for structured peer-to-peer overlays,” in *Proc. the 2nd International Workshop on Peer-to-Peer Systems (IPTPS’03)*, 2003.
[11] S. S. Laurent, J. Johnston, and E. Dumbill, “Programming web services with xml-rpc,” 2001.
[12] “BIND (Berkeley Internet Name Domain).” [Online]. Available: http://www.isc.org/software/bind